IT kompakt

Manfred Brill

Virtual Reality kompakt

Entwicklung von immersiver Software

Springer Vieweg

IT kompakt

Die Bücher der Reihe „IT kompakt“ zu wichtigen Konzepten und Technologien der IT:

- ermöglichen einen raschen Einstieg,
- bieten einen fundierten Überblick,
- eignen sich für Selbststudium und Lehre,
- sind praxisorientiert, aktuell und immer ihren Preis wert.

Manfred Brill

Virtual Reality kompakt

Entwicklung von immersiver Software

Springer Vieweg

Manfred Brill
Informatik und Mikrosystemtechnik
Hochschule Kaiserslautern
Zweibrücken, Deutschland

ISSN 2195-3651 ISSN 2195-366X (electronic)
IT kompakt
ISBN 978-3-658-41244-9 ISBN 978-3-658-41245-6 (eBook)
https://doi.org/10.1007/978-3-658-41245-6

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über https://portal.dnb.de abrufbar.

Planung/Lektorat: Petra Steinmueller
Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer Fachmedien Wiesbaden GmbH und ist ein Teil von Springer Nature.
Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Das Papier dieses Produkts ist recyclebar.

Für Maximilian.
Lass uns Lummerland bauen!

Vorwort

„Die Zeit ist reif für die Entwicklung von produktiven Anwendungen“ – dieser Satz aus dem Vorwort von „Informatik im Fokus: Virtuelle Realität“ [1] aus dem Jahr 2009 gilt heute in noch viel größerem Maße. Die Entwicklungen, die mit der Vorstellung der ersten Oculus Rift ihren Anfang nahmen, haben dazu geführt, dass VR den Weg aus Forschungs-Abteilungen und Hochschulen in unsere Wohnzimmer gefunden hat. Waren Studierende in einer Lehrveranstaltung zum Thema VR vor einigen Jahren noch neugierig darauf, VR erleben zu können, vergleichen sie heute die Labor-Hardware mit der Ausstattung zu Hause.

Die Software-Entwicklung für VR, darum geht es in diesem Buch, ist nach wie vor eine Herausforderung. Hier treffen sehr viele Teilgebiete der Informatik aufeinander. Unmöglich dies alles in ein einziges Buch aufzunehmen. Wo fängt man an, was kann man voraussetzen? Je nach Blickwinkel der Leserinnen und Leser fehlt garantiert etwas in diesem Buch. Es fiel nicht immer leicht, eine Entscheidung zu treffen, was jetzt wo und wie weggelassen werden soll. Ich hoffe, dass das Ergebnis trotzdem von Nutzen ist. Zu diesem Buch gibt es die Website mbrill.github.io/VRKompakt und das GitHub-Repository github.com/MBrill/VRKompakt mit Unity-Projekten, Lösungen und weiteren Informationen.

An dieser Stelle möchte ich mich bei allen bedanken, die zum Entstehen dieses Buchs beigetragen haben. Der Dank geht insbesondere an Frau Petra Steinmüller aus dem Springer-Verlag für die Idee, dieses Buch zu realisieren. Die Diskussionen über den Fokus und die Zielgruppe haben mir sehr geholfen, die

notwendigen Entscheidungen zu treffen. Hervorragend und wie gewohnt professionell war die Zusammenarbeit mit allen Mitarbeitern des Springer-Verlags die zum vorliegenden Ergebnis beigetragen haben. Herzlichen Dank dafür an dieser Stelle!

Nicht zu vergessen meine Familie, bei der ich mich an dieser Stelle wieder einmal für das Verständnis bedanke, Wochenenden und Abende damit zu verbringen, ein Buch zu verfassen. Vielen Dank für die Geduld und die Unterstützung in dieser Zeit!

Saalstadt, Deutschland Manfred Brill
im Mai 2023

Literatur

1. Brill, M.: Virtuelle Realität. Springer (2009)

Inhaltsverzeichnis

Einleitung 1

„If real is what you can feel, smell, taste and see, then 'real' is simply electrical signals interpreted by your brain. [...] Unfortunately, no one can be told what the Matrix is. You have to see it for yourself“,

„Morpheus“ in „The Matrix“ [14].

Zusammenfassung

Wir beginnen dieses Buch mit dem Versuch, den Begriff Virtuelle Realität zu erklären und einzuordnen. Dazu bauen wir ein abstraktes Modell auf, das wir im weiteren Verlauf des Buchs konkretisieren werden. Für die Software-Entwicklung setzen wir Unity ein, das wir kurz vorstellen. Ein Überblick über die folgenden Kapitel, typografische Konventionen und Angaben zu Quellen schließen das Kapitel ab.

1.1 Virtuelle Realität

Wir beginnen dieses Buch mit der Aufgabe zu erklären, was hinter den Begriffen *Virtuelle Realität* oder *Virtual Reality* eigentlich steckt. Wir werden meist die Abkürzung *VR* dafür verwenden.

M. Brill, *Virtual Reality kompakt*, IT kompakt,
https://doi.org/10.1007/978-3-658-41245-6_1

Eine Antwort auf diese Frage in Text und Bild ist eine Herausforderung. Am Sinnvollsten ist es, eine VR-Anwendung auszuführen und sich selbst eine Vorstellung davon zu machen.

Menschen nehmen die Realität mit Hilfe der Sinnesorgane war. Wir verzichten hier bewusst auf die philosophische Fragestellung, die sich sofort daraus ergibt und die ausgehend von Platons Höhlengleichnis die Frage diskutiert, was diese Realität jetzt wirklich ist. Im Folgenden verstehen wir unter *Realität* die physikalische Realität, in der sich die Anwender bewegen. Der Begriff „virtuell" stammt aus dem Französischen und kann mit „scheinbar" übersetzt werden. Wir sprechen von virtuellen Laufwerken – eine Datei-Ablage, die sich wie ein Speicher-Medium in unserem Rechner verhält, in Wirklichkeit aber nicht ist. Die Funktionalität ist identisch, wir nehmen das virtuelle Laufwerk wie eine physikalische Hardware wahr. Den Begriff Virtual Reality hat wahrscheinlich Jerome Lanier als Teilnehmer auf einer SIGGRAPH-Podiumsdiskussion zum ersten Mal verwendet [2].

Ersetzen wir die Sinnes-Reize, mit denen die physikalische Realität wahrgenommen wird, durch künstlich hergestellte Signale, sind wir in der Lage, eine andere, eine virtuelle, Realität zu vermitteln. Ob dieser Ansatz einer virtuellen Realität erfolgreich ist, hängt letztendlich von den Vorgängen im menschlichen Gehirn ab. Menschen sind bereit sich auf solche Illusionen einzulassen. Das machen sich schon Buch, Theater oder Film zu Nutze. Samuel Coleridge hat dies 1817 mit dem Begriff „willing suspension of disbelief", in Deutsch „willentliche Aussetzung der Ungläubigkeit", charakterisiert [16]. Menschen sind bereit die Vorgaben eines fiktionalen Werks vorübergehend zu akzeptieren, auch wenn diese eigentlich unmöglich erscheinen. Wir wissen, dass James Bond ein britischer Geheimagent ist und sicher Englisch spricht – das stört uns bei einem synchronisierten Film überhaupt nicht.

Die technische Realisierung solcher Systeme ist keine ganz neue Entwicklung des letzten Jahrzehnts, auch wenn uns das einige Hersteller von VR-Hardware glauben machen wollen. Der Informatik-Pionier Ivan Sutherland hat in *The Ultimate Display* [10] schon in den sechziger Jahren des letzten Jahrhunderts ein

solches System, das mit Hard- und Software eine virtuelle Realität erzeugte, beschrieben und realisiert.

Akzeptieren die Benutzer eines VR-Systems die Stimuli als real, bezeichnen wir dies mit dem Begriff *Immersion*, im Englischen finden wir darüber hinaus die sehr treffende Beschreibung „being there“. Wir rufen ein Gefühl der *Präsenz* in der künstlich erzeugten Umgebung hervor. Die Wahrnehmung alleine wird für diese Präsenz nicht ausreichen. Wir müssen uns in der virtuellen Welt orientieren und fortbewegen können. Und wir interagieren mit Objekten in dieser virtuellen Umgebung. Dabei ist es möglich, dass die Objekte den uns bekannten Naturgesetzen folgen, oder gänzlich anderes Verhalten an den Tag legen. Maßgeblich ist immer, ob diese Interaktion die Immersion stört oder eher unterstützt. Wir müssen dabei davon ausgehen, dass die Reaktion auf Interaktionen in Echtzeit erfolgt. Gelingt dies nicht wird die Immersion sehr schnell empfindlich gestört.

Inzwischen finden wir neben dem Begriff Virtual Reality die *Erweiterte Realität* oder *Augmented Reality*, kurz *AR*. Hier hilft das von Milgram und Kishino entwickelte „Realitäts-Virtualitäts-Kontinuum“ [9], das wir in Abb. 1.1 finden. Die physikalische Realität und die virtuelle Realität stellen die beiden Antipoden dar. Die Übergänge zwischen den Realisierungen dieser „Realitäten“ sind stufenlos, dies soll der Begriff „Kontinuum“ ausdrücken. Wann sprechen wir von AR, wann von VR? Sind die Ausgaben einer Anwendung nur zu einem kleineren Teil durch den Computer erzeugt, können wir sie der erweiterten Realität zuordnen. Neben dem Begriff der Mixed Reality wird ein weiterer Oberbegriff verwendet. Mit *Extended Reality* oder *Cross Reality*, abgekürzt *XR*, werden alle Technologien wie MR, AR oder VR zusammengefasst. Wie schon der Titel dieses Buchs nahelegt,

Abb. 1.1 Mixed Reality, das Realitäts-Virtualitäts-Kontinuum nach Milgram und Kishino

werden wir uns mit der Entwicklung von VR-Anwendungen auseinandersetzen.

Für die Realisierung einer VR-Anwendung setzen wir ein VR-System aus Hard- und Software ein. In Abb. 1.2 ist eine Darstellung eines solchen Systems zu sehen. Die Ultima Ratio eines VR-Systems ist selbstverständlich der vollständige Ersatz aller Sinnesreize durch synthetisch erzeugte Signale. Das wird technisch häufig nicht möglich sein. Darüber hinaus dürfte es auch zukünftig schwierig sein, Stimuli für den Geschmack oder die menschliche Schmerzempfindung umzusetzen. Es darf mit Recht bezweifelt werden, dass solche technischen Möglichkeiten von großem Nutzen wären.

Die Bedienung eines VR-Systems, das wir für die Ausführung einer Anwendung aus dem Mixed Reality-Kontinuum einsetzen, unterscheidet sich stark von der Art und Weise wie wir mit einer Desktop-Anwendung oder einer App auf einem mobilen Endgerät umgehen. Eingaben auf Grund von Interaktionen mit virtuellen

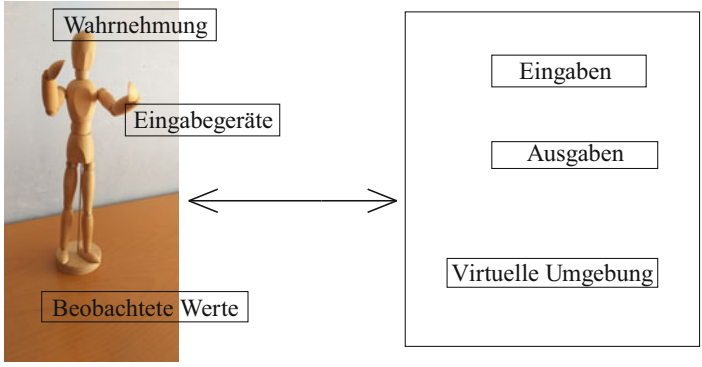

Abb. 1.2 Die Anwender bewegen sich in der physikalischen Realität, nehmen die virtuelle Umgebung wahr und interagieren mit ihr. Auf einem oder mehreren Computern existiert eine Beschreibung der virtuellen Welt, die von den Nutzern bewusst und unbewusst erzeugten Daten werden verarbeitet und die Daten für die Wahrnehmung der virtuellen Realität werden erzeugt

Gegenständen werden an die Computer im VR-System gesendet und verarbeitet. Neben bewusst von den Anwendern ausgelösten Ereignissen werden unbewusst erzeugte Daten erfasst und übertragen. Wir sprechen von *beobachteten Werten*. Die Änderung der Blickrichtung, eine Handbewegung oder die Fortbewegung in der physikalischen Realität wird mit Hilfe von Sensoren erfasst und an die Anwendung auf dem Computer übertragen. Die Verfolgung von Positionen und Orientierungen wird bei VR-Systemen als *Tracking* bezeichnet. Die Anwendung verwendet die im Computer vorhandene Beschreibung der virtuellen Welt und reagiert auf die eingehenden Daten, um die von den Anwendern wahrgenommene virtuelle Umgebung in Echtzeit anzupassen. Die virtuelle Welt kann im Maßstab 1 : 1 gestaltet sein. Ein VR-System macht es aber auch möglich, uns wie Gulliver nach Liliput oder Brobdingnag zu begeben. Wir können die Gesetze der Physik übernehmen, oder eigene Gesetzmäßigkeiten realisieren. Die Gestaltung der virtuellen Welt, die unbewusst entstehenden Eingaben und die Reaktion darauf sind Schlüssel-Elemente für die Immersion und Präsenz. Sie machen den Reiz der virtuellen Realität aus.

Dass die menschliche Wahrnehmung zu einer See- oder allgemein Reise-Krankheit führen kann, ist schon lange bekannt. Man weiß, dass die Wahrnehmung von niederfrequenten Vibrationen ein Grund für das Auftreten dieser Syndrome sein kann. Schon früh wurde beim Einsatz von Flugsimulatoren berichtet, dass die Anwender über Unwohlsein klagten. Dies wird als *Simulatorkrankheit* bezeichnet. Das sollten wir bei der Gestaltung und Realisierung einer VR-Anwendung immer berücksichtigen, denn auch bei der Nutzung von VR-Anwendungen werden solche Symptome beobachtet. Dafür finden wir den Begriff *Cybersickness*. Die Ursachen für das Auftreten dieses Syndroms sind noch nicht ganz verstanden. Menschen nehmen Bewegungen und Beschleunigung mit Hilfe des Vestibularsystems wahr. Bewegen sich die Anwender eines VR-Systems überhaupt nicht, nehmen aber visuell eine schnelle oder abrupte Bewegung wahr kann dieser Widerspruch zu Cybersickness führen. Eine gute Darstellung dieses Aspekts von VR-Anwendungen finden wir in [3].

1.2 Software-Entwicklung für die virtuelle Realität

Wie in vielen anderen Bereichen der Informatik, stellt sich auch bei der Realisierung von VR-Anwendungen sofort die Frage, wie wir die Software dafür effizient und wirtschaftlich realisieren können. Natürlich können wir solche Anwendungen nativ implementieren. Damit ist gemeint, dass wir die für die jeweilige Plattform am besten geeignete Programmiersprache, das passende Grafik- und Audio-API und andere Schnittstellen einsetzen. Wir stehen vor dem Grundproblem der Software-Entwicklung. Wie können wir einen möglichst hohen Grad an Wiederverwendung von Software-Komponenten erreichen? Wie stellen wir sicher, dass unsere Version für Plattform X die Ressourcen dort genauso optimal ausnutzt wie auf Plattform Y? Von den Problemen, für jede dieser Umgebungen exzellente Software-Entwickler zu finden gar nicht zu reden.

Als Lösung haben sich inzwischen Entwicklungsumgebungen etabliert, die nach dem Prinzip „Code once – run everywhere“ arbeiten. Wir verwenden Programmiersprachen wie Java, C++ oder C# und erstellen damit eine Anwendung für viele verschiedene Plattformen. Als Grafik-API werden Vulkan/OpenGL oder Direct3D eingesetzt. Die Anpassung an die Laufzeit-Umgebung wird für uns von der Entwicklungsumgebung durchgeführt. Die beiden wichtigsten Vertreter dieser Philosophie im Bereich der Spiele- und VR-Software sind zur Zeit Unreal [4] und Unity [13]. Beide Lösungen haben Vor- und Nachteile, die an dieser Stelle nicht diskutiert werden sollen. Am Sinnvollsten wäre es dieses Buch so zu schreiben, dass wir beide Systeme im Einsatz zeigen. Der zur Verfügung stehende Platz spricht allerdings gegen diesen sehr naheliegenden Ansatz. Für diese Monographie fiel die Entscheidung auf Unity. Die Entscheidung fiel nicht leicht. Man kann sehr lange und intensiv über Pro und Contra der beiden Angebote diskutieren. Wichtig ist der Hinweis, dass wir alles was wir in diesem Buch betrachten in Unreal genauso gut umsetzen können.

Dieses Buch ist keine Einführung in Unity, Grundkenntnisse werden vorausgesetzt. Wir konzentrieren uns darauf die für die Entwicklung von Virtual Reality-Anwendungen wichtigen Konzepte zu behandeln. Für den Einstieg in die Anwendungsentwicklung mit Unity gibt es bei Bedarf eine große Menge von Monographien und Online-Ressourcen. Von Unity selbst gibt es die Plattform Unity Learn [12]. Das Buch setzt Kenntnisse in objekt-orientierter Software-Entwicklung mit C# [5] voraus . Für Leser, die mit Java vertraut sind, gibt es von Microsoft das Cheat-Sheet [7], für den Übergang von C++ zu C# finden wir einen guten Überblick in [6]. Für Unity gibt es regelmäßige Updates und Beta-Versionen. Alle Beispiele im Buch und auf der Website zum Buch wurden mit 2021.3.29f1 LTS realisiert.

1.3 Aufbau des Buches

In Kap. 2 beginnen wir mit der Entwicklung von interaktiven Anwendungen mit Unity. Wir bauen ein Projekt auf, das wir mit Variationen im Buch einsetzen werden. Damit wir uns auf die behandelten Themen konzentrieren können verzichten wir darauf, die Szenen in den Projekten mit Assets zu überladen, die das Nachvollziehen der vorgestellten Themen nur unnötig erschweren würden.

Wir setzen uns damit auseinander, wie wir in Unity Interaktionen realisieren. Anschließend betrachten wir das Erstellen von Protokollen, mit der Klasse `Debug`, aber auch mit Frameworks, die im Umfeld von C# eingesetzt werden. Protokollierung hilft nicht nur während der Software-Entwicklung, sondern auch bei der Evaluation einer VR-Anwendung oder der Fehlersuche. Wir verbinden dieses Thema mit der Darstellung von Kollisionen und Raycasting in Unity. Testen ist ein unverzichtbarer Teil der professionellen Software-Entwicklung. Wir betrachten das Unity Test Framework und zeigen den Einsatz der Tests bei der Realisierung der World-in-Miniature Technik.

In Kap. 3 kommen wir zum Hauptthema dieses Buchs, der Entwicklung von VR-Anwendungen. Bevor wir zur Software-Entwicklung kommen, betrachten wir einige Grundlagen, die wir in den konkreten Software-Lösungen wiederfinden. Seit einiger Zeit wird der Versuch unternommen, mit OpenXR eine neutrale Programmierschnittstelle für die Realisierung von XR-Anwendungen zu definieren. Auch Unity unterstützt diese Initiative. Aus der Menge der Packages die für die VR-Entwicklung eingesetzt werden können, haben Unity XR und VIVE Input Utility den Weg in diesem Buch gefunden. Das Vorgehen für die Verwendung von anderen Packages wie zum Beispiel MiddleVR [8] ist sehr ähnlich. Es sollte keine großen Probleme bereiten die vorgestellten Vorgehensweisen zu übertragen. Ein Schwerpunkt des Kapitels ist die Realisierung von immersiven Benutzungsoberflächen für die Systemsteuerung, die Auswahl, die Manipulation von Objekten und der Fortbewegung in einer virtuellen Umgebung.

Wer VR-Anwendungen realisieren möchte, hat mit relativ großer Wahrscheinlichkeit Zugriff auf entsprechende Hardware. Dass VR-Hardware vorhanden ist, wird in diesem Buch jedoch nicht vorausgesetzt. Die vorgestellten Lösungen bieten die Möglichkeit eines Simulators. Die Fehlersuche im Quelltext ist eine Herausforderung, wenn wir ein Head-Mounted Display tragen. Solche Simulatoren bieten also nicht nur die Möglichkeit, dieses Buch ohne Hardware zu nutzen – sie stellen eine wichtige Komponente in der Software-Entwicklung von immersiven Anwendungen dar. Ein Simulator macht die Anwendungsentwicklung unabhängig von den Grenzen und Eigenschaften spezieller Hardware.

Natürlich reicht es nicht ein Buch zu lesen, wir müssen die Theorie und die Praxis selbst nachvollziehen. Dazu finden wir Abschnitte mit der Überschrift „Do it yourself“. Lösungshinweise dazu finden wir im Anhang. Die Website zum Buch [1] enthält die Unity-Projekte zu den Lösungen und weitere Beispiele in Form von GitHub Pages und einem Repository. Auf den Seiten zum Buch finden wir Tipps, Links, Texte und die unvermeidlichen Errata.

1.4 Typografische Konventionen

Um die Lesbarkeit dieses Buchs zu erhöhen, werden einige typographische Konventionen eingeführt. Menu-Befehle oder Ausgaben auf der Konsole werden in `Schreibmaschinenschrift` dargestellt. Damit soll es erleichtert werden, diese Text-Teile und die dazu gehörigen Beschreibung im Text unterscheiden zu können. Adressen im WWW, Dateien und Verzeichnisse, aber auch Tastatureingaben werden ebenfalls in `Schreibmaschinenschrift` dargestellt. Bei allgemeinen Angaben zu Verzeichnissen verwenden wir eine Schreibweise wie auf UNIX-Systemen: `Assets/Scenes`. Pfade auf einem Windows-Rechner geben wir mit der entsprechenden Syntax an: `C:\local\logs`.

Quelltexte, meist in der Sprache C#, werden häufig nur in Ausschnitten im Text aufgenommen:

Beispiel für einen Quelltext

```
public HandRole MainHand = HandRole.RightHand;
public ControllerButton TheButton = Controller
   Button.Trigger;
```

◀

Variablen, Klassen oder andere Bestandteile einer Implementierung werden im Text ebenfalls mit diesem Font dargestellt: `MonoBehaviour`.

Vektoren in mathematischen Formeln werden als fettgedruckte Kleinbuchstaben geschrieben: das Symbol $\mathbf{v}$ steht für einen Vektor. Punkte in einem Koordinatensystem schreiben wir als Großbuchstaben: P. Ein Vektor, der im Punkt P beginnt und im Punkt Q endet schreiben wir als $\mathbf{PQ}$, $d(P, Q)$ steht für den euklidischen Abstand der Punkte. Die euklidische Länge eines Vektors schreiben wir als $||\mathbf{x}||$.

1.5 Lizenzen und Markenzeichen

Um die Realisierung mit Hilfe von Unity darzustellen finden wir Ausschnitte des Unity Editors wie in Abb. 1.3 links. Räumliche Ansichten der Unity-Szenen wie in Abb. 1.3 rechts wurden mit Hilfe der Vorschau in Unity realisiert und abgespeichert. Diese Abbildungen wurden alle vom Autor erstellt.

Die Darstellungen in diesem Buch halten sich an die von Unity publizierten Richtlinien: *These materials are not sponsored by or affiliated with Unity Technologies or its affiliates. „Unity" is a trademark or registered trademark of Unity Technologies or its affiliates in the U.S. and elsewhere* [11].

Das Package VIVE Input Utility [15] wird mit freundlicher Genehmigung von HTC verwendet: *Vive Input Utility (VIU) is copyright 2016-2023, HTC Corporation. All rights reserved.*

In den Unity-Projekten wird eine Textur auf der Basis der Bitmap eingesetzt, die in Abb. 1.4 zu sehen ist. Diese Bitmap zeigt das Audimax am Campus Zweibrücken der Hochschule Kaiserslautern. Wir verwenden diese Textur in den Unity-Projekten und den dazugehörigen Abbildungen mit freundlicher Genehmigung der Hochschule Kaiserslautern.

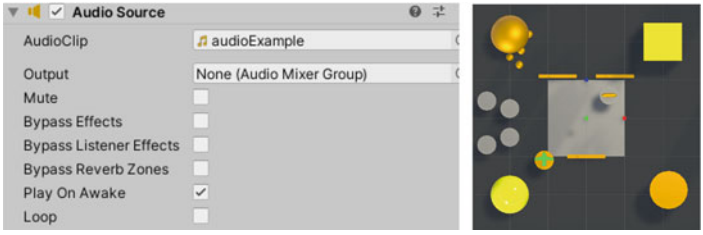

Abb. 1.3 Links: Ausschnitt des User Interface des Unity Editors. Rechts: Beispiel der Darstellung einer Unity-Szene. Alle Abbildungen wurden vom Autor erstellt